REMARKS

Applicant is in receipt of the Advisory Action mailed June 3, 2008. Claims 1, 10, and 18-21 have been amended. Claims 1 and 6-23 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Section 103 Rejections

Claims 1 and 6-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Deutscher et al (2004/0001106, "Deutscher") in view of Hampapuram et al. (2004/0221262 A1, "Hampapuram"). Applicant respectfully traverses the rejection.

Amended claim 1 recites:

1. A computer accessible memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, wherein, during execution of the program, the program instructions are executable by a processor to perform:

displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression;

receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression.

Applicant respectfully submits that Deutscher fails to disclose, **displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code**, as recited in claim 1.

As discussed in the previous Response, Deutscher is directed to a multimedia presentation production system in which presentation data are separated from presentation logic (see, e.g., paragraph [0008]). More specifically, per [0009], Deutscher discloses a tool which is

> ...essentially an intelligent visual data editor for making rich media presentations. Upon publishing from the tool, the required elements are organized and edited to contain the proper metadata in a single publishing directory. More particularly, **the tool consists of graphic user interfaces for easily entering the data associated with the rich media presentation. The tool is also designed to make it simple and easy to edit the data of the underlying schema**. (*emphasis added*)

As described in [0017]-[0018], among the various data entry graphical user interfaces (GUIs) disclosed are "data entry grids", which are displayed in a presentation tool window. As [0018] reads in pertinent part:

> ...**the data entry grids allow the user to enter the information into the presentation data file and the master track media file that is needed to drive the timeline of the presentation. The first of these grids**, which is displayed by default once the master track program has been imported, **is the scripts grid**. The scripts grid is where the user enters events called script commands that will be triggered during the playback of the presentation. (*emphasis added*)

The Office Action equates Deutscher's scripts grid with Applicant's claimed source code of an executing program, which Applicant respectfully submits is not correct, since, as Deutscher explains above, and as described in more detail in Deutscher's section 2.6.1 ([0137] – [0144]), the scripts grid is:

> **essentially a scheduling table** where the user enters events (sometimes referred to as script commands) that will be triggered during the playback of the presentation (*emphasis added*)

Note that the entered data in the various data grids (including the scripts grid) are saved to the presentation data file, which is then referenced by the presentation system to present the multimedia presentation. A scheduling table is *not* source code that is compilable to generate an executable program.

Applicant submits that this is a significant different, since "setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression" is considerably more complex for compiled source code than for the grid in Deutscher.

In direct contrast to Deutscher's script grid, as one of skill in the programming arts readily understands, "source code" is a term of art in the programming domain and refers specifically to program instructions in a programming language that are compiled to produce executable code that may be run on a processor. This is clarified in the amended independent claims by the limitation: "wherein the executing program was compiled from the source code".

Clearly, the cited script grid of Deutscher is not source code of a program, and more specifically, is not source code for an *executing* program, but, as described in paragraph 137, is a scheduling table, and more specifically, an editing tool for editing and providing data to a data file, which is then used by a presentation application to present a multimedia presentation. Nor does Deutscher describe any of the data grids as source code for a(n executing) program. Note, for example, that per paragraph 137, the program that is paused to allow for Deutscher's scripts grid editing is "a video or audio program that is designated as the master track", not an executable program for which the scripts grid is source code.

The Advisory Action asserts that Deutscher's script grid and Applicant's source code "perform the same functionality", stating that "Source code is read by the computer to perform the final outcome of the presentation. Script grid is read by the computer to perform the final outcome of the presentation. Each may be written, composed and used differently but essentially they perform the same function to produce a same end result, that they are both computer instructions for performing desired entered task by the computer to output a desired result". This is not a proper analysis or characterization of the technical content of claim 1.

Applicant respectfully reminds the Examiner that *every word in the claim* is to be given consideration, and that the terms of the claim are to be interpreted in light of the Specification. More specifically, terms of art should not be re-interpreted contrary to the Specification.

Applicant submits that one of ordinary skill in the art would readily understand that a scheduling table (Deutscher's description of the script grid) used by an executing program to manage multi-media presentations does not "perform the same function" as source code that is compiled to generate the executable program. Note that substituting either of these elements for the other in their respective inventions would render the inventions inoperable, and so they clearly do *not* perform the same function, and are not equivalent.

Applicant notes that compiled source code has significant speed and efficiency improvements over the scheduling table in Deutscher. Also, claim 1 allows the expression in the program to be set to the new value while the program continues execution in accordance with the new value of the expression. This is a significantly more difficult problem in compiled source code than the method taught in Deutscher.

Thus, for at least the above reasons, Applicant submits that Deutscher fails to teach or suggest these features of claim 1.

Applicant respectfully notes that since the cited art fails to teach "displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code", the cited art also does not, and cannot, teach "receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program", nor "in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression", nor "receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression", nor "setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression", as claimed.

For example, the cited art nowhere discloses **receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;** nor **in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression**, as recited in claim 1.

As a careful reading of Hampapuram and Deutscher makes clear, neither reference discloses the claimed hover invoked functionality as claimed. As discussed above, Deutscher discloses editing a script grid, which, as Deutscher defines it, is "essentially a schedule table", which is not source code as defined in claim 1. Nor is Deutscher's editing of the script grid performed during execution of the program. Applicant further notes that Deutscher teaches displaying the pop-up window in response to user input, specifically, double-clicking on the item to be edited, whereas in claim 1, the GUI element is automatically displayed in response to simply hovering the mouse cursor over the expression.

Nor does Hampapuram remedy these deficiencies of Deutscher. For example, per Hampapuram's Abstract, the macro expansions are processed by Hampapuram's tool during the build process of a project ("during a build of a programming project's source files"), where this processing operates to collect and record the macro expansion information into an output file or database. The tool then uses the recorded information to "display the macro expansions in a graphical user interface of the tool, such as for source browsing or viewing static analysis". Applicant notes that "source browsing" and "viewing static analysis" are not run-time operations, and are *nowhere described as being performed while the program is executing*.

Applicant respectfully notes that "expanding a macro" is not at all the same as automatically displaying a GUI element proximate to the expression (over which the user hovers a mouse cursor *during execution of the program*), where the GUI element includes a value of the expression. More particularly, Hampapuram's GUI does not display the value of an expression during execution of the program, but rather simply displays a macro expansion statically, i.e., *not* at runtime.

Thus, the cited art fails to teach or suggest these features of claim 1.

Nor does the cited art disclose **receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression**, as recited in claim 1.

As discussed above, neither Deutscher nor Hampapuram discloses displaying a value of an expression in source code of an executing program, i.e., at runtime. Nor do Deutscher and Hampapuram disclose modifying the value of such an expression in the source code of an executing program, *where the program continues execution in accordance with the new value of the expression*. Note, for example, that Deutscher's editing of the script grid neither modifies an expression in source code as claimed, nor is performed during execution of the program. Similarly, Hampapuram's macro expansions do not teach or suggest modifying the value of an expression in the source code of an executing program, where the program continues execution in accordance with the new value of the expression, and is similarly not performed during execution of the program.

The Office Action asserts that it would have been obvious to one of ordinary skill in the art to combine Hampapuram and Deutscher because "one of ordinary skill in the art would recognize the program being used in the system of Deutscher does not have to be program specific for the function ality of a pop-up control and that the pop-up control could work in any program environment (e.g., a debugger)", and further asserts that "the combination of Hampapuram into Deutscher would yield the predictable result of having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system [*sic*]".

Applicant respectfully disagrees.

The Examiner characterizes the functionality taught by a combination of Hampapuram and Deutscher as "having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system". However, Applicant respectfully notes that there is a substantial difference between simply changing a value in a static editor as taught by Deutscher or expanding a

macro in a source code browser or viewer as taught by Hampapuram, and dynamically modifying a value of an expression during execution of a program as claimed, i.e., via user input to a tooltip invoked via hovering the mouse over the expression during runtime, as claimed. For example, note that Deutscher edits text in the script grid, e.g, script type, script parameter, which is subsequently referenced by the program upon execution when the program accesses the presentation file to which the script grid's changes have been copied. This is not a dynamic process performed at runtime, and is very different from Applicant's dynamic modification of a value of an expression during runtime, where the executing program continues to execute using the modified value.

Nor does Hampapuram's GUI facilitate editing the value of an expression during execution of the program, but rather simply displays a macro expansion statically, i.e., *not* at runtime.

Applicant respectfully submits that since neither reference discloses these features, one of skill in the art would not be compelled to combine them in an attempt to generate Applicant's invention, and so Applicant submits that a proper motivation to combine has not been provided, and thus, Hampapuram and Deutscher are not properly combinable to make a prima facie case of obviousness.

Moreover, even were Hampapuram and Deutscher properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as claimed.

Thus, for at least the reasons provided above, Applicant submits that the cited art of Deutscher and Hampapuram, taken singly or in combination, fails to teach or suggest all the features and limitations of claim 1, and so claim 1, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claims 18-21 each includes similar limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons discussed above, Applicant submits that claims 18-21, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. For example, nowhere does the cited art disclose **wherein the GUI element tooltip comprises: a first portion, operable to display the value of the expression, wherein the first portion is further operable to receive the second user input modifying the value; and a second portion, operable to display non-editable information related to the expression**, as recited in claim 10.

Deutscher's pop-up window only displays editable fields for editing the script information, and Hampapuram's pop-up window displays a macro expansion, and does not allowing editing of the expansion. Thus, the references fail to disclose these features, and so claim 10, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable..

Applicant also asserts that numerous other ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 103 rejection of claims 1 and 6-23 is earnestly requested.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-82801/JCH.

Also filed herewith are the following items:

☒ Request for Continued Examination

☒ Information Disclosure Statement

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-06-20 JCH/MSW